Documentation for math2.h and math2.c

Steven Andrews, © 2004
See the document "LibDoc" for general information about this and other libraries.

```
#ifndef __math2_h
#define __math2_h

#define PI 3.14159265358979323846
#define SQRT2 1.41421356237
#define SQRT2PI 2.50662827462
#define SQUARE(X) ((X)*(X))
#define QUADPLUS(A,B,C) ((-(B)+sqrt((B)*(B)-4*(A)*(C)))/(2*(A)))
#define QUADMINUS(A,B,C) ((-(B)-sqrt((B)*(B)-4*(A)*(C)))/(2*(A)))

int sign(float x);
int signD(double x);
float sinc(float x);
float box(float x);
float bessj0(float x);
float bessj1(float x);
float gauss(float x,float mean,float sd);
float gammaln(float x);
float gammp(float a,float x);
float erfn(float x);
float erfnc(float x);
float betaln(float x1,float x2);
int iseven(int x);
int is2ton(int x);
int isinteger(float x);
int next2ton(int x);
float factorial(int n);
float choose(int n,int m);
int minus1to(int x);
int intpower(int n,int p);
int gcomdiv(int m,int n);
float hermite(float x,int n);
float reflect(float x,float lo,float hi);
float inversefn(float (*fn)(float),float y,float x1,float x2,int n);
double quadpts2paramsD(double *x,double *y,double *abc);
double fouriersumD(double *a,double *b,int n,double l,double x);
void SetHillParam(float *hp,float a,float e,float n);
float HillFn(float *hp,float x);
void HillFnCompose(float *hp1,float *hp2,float *hptot);
void HillFnComposeNF1(float *hp1,float *hp2,float *hpf1,float *hpf12);

#endif
```

Requires: <math.h>,<stdlib.h>, <float.h>, "math2.h"
Example program: LibTest.c

History: Modified 9/2/98. Modified again 1/00 and again 8/00. Works with Metrowerks C. Documentation updated 10/19/01. Updated 1/8/02. Fully tested. Added `isinteger` 2/20/02. Modified `next2ton` slightly 10/28/03. Added QUADPLUS and QAUDMINUS and renamed sqr macro to SQUARE 4/16/04. Added `quadpts2params` 9/20/05. Added `signD` 10/19/05. Added `fouriersumD` 12/16/05. Added the Hill function functions 11/28/06.

This library file complements the standard math.h library with many useful functions. No effort is made to check for overflow problems (i.e. routines may return `Inf`, or comparable error codes). A couple routines are copied directly from *Numerical Recipes*. A routine called `bessj1int` was removed 10/18/01 since it was inferior to `bessj1`; the inputs and outputs are identical.

Summary of math functions

Name	Domain	Output
SQUARE	$(-\infty,\infty)$	x^2
QUADPLUS	$(-\infty,\infty)^3$	positive real root of quadratic eqn. if exists, or `nan` if not
QUADMINUS	$(-\infty,\infty)^3$	negative real root of quadratic eqn. if exists, or `nan` if not
sign	$(-\infty,\infty)$	+1, 0, or –1 for a positive, zero, or negative argument
sinc	$(-\infty,\infty)$	sin(x)/x
box	$(-\infty,\infty)$	1 for $-1 \leq x \leq 1$ and 0 elsewhere
bessj0	$(-\infty,\infty)$	J0 Bessel function using *Numerical Recipies* routine
bessj1	$(-\infty,\infty)$	J1 Bessel function using *Numerical Recipies* routine
gauss	$(-\infty,\infty)$	Normalized Gaussian*
gammaln	$(-\infty,\infty)$	Natural log of absolute value of gamma function*
gammp	$(0,\infty),[0,\infty)$	Incomplete gamma fn., P(a,x); –1 for input out of domain
erfn	$(-\infty,\infty)$	Error function
erfnc	$(-\infty,\infty)$	Complementary error function
betaln	$(-\infty,\infty)^2$	Natural log of the beta function
iseven	$(-\infty,\infty)$	1 if even, 0 if odd
is2ton	$(-\infty,\infty)$	1 if x is an integer power of 2, 0 if not
isinteger	$(-\infty,\infty)$	1 if x is an integer, 0 if not
next2ton	$(-\infty,\infty)$	Next higher power of 2, 1 if x=0, or 0 if x<0
factorial	$[0,\infty)$	n!, by computation of products; returns 1 for n<0
choose	$[0,\infty),[0,n]$	n choose m
minus1to	$(-\infty,\infty)$	$(-1)^x$
intpower	$(-\infty,\infty)^2$	n^p, as an integer, p<0 returns 0
gcomdiv	$(-\infty,\infty)^2$	greatest common divisor using a varient of Euler's method*
hermite	$(-\infty,\infty),[0,\infty)$	Hermite polynomial by recursion; returns 0 for n<0
reflect	$(-\infty,\infty)$	Reflects *x* into range between `lo` and `hi` using recursion
HillFn	$[0,\infty)$	Hill function with variable x and parameters hp.

Defined constants

Name	Value	Meaning
PI	3.14159265358979323846	π
SQRT2	1.41421356237	$\sqrt{2}$
SQRT2PI	2.50662827462	$\sqrt{(2\pi)}$

Macro functions

```
#define SQUARE(X) ((X)*(X))
```
     Returns X^2.

```
#define QUADPLUS(A,B,C) ((-(B)+sqrt((B)*(B)-4*(A)*(C)))/(2*(A)))
```
     QUADPLUS returns the positive real root of the quadratic equation for coefficients A, B, and C. If A and C have the opposite sign, then a positive root exists and is greater than 0.

```
#define QUADMINUS(A,B,C) ((-(B)-sqrt((B)*(B)-4*(A)*(C)))/(2*(A)))
```
     QUADMINUS returns the negative real root of the quadratic equation for coefficients A, B, and C. If A and C have the opposite sign, then a negative root exists and is less than 0.

Functions

Many functions are only described in the summary table shown above.

```
float gauss(float x,float mean,float sd);
```
     This returns a Gaussian with mean mean and standard deviation sd. sd may not be 0 but a negative sd yields a negative Gaussian.

```
float gammaln(float x);
```
     This uses direct summation for integer and half integer values of x, recursion for other negative values, and a formula from *Numerical Recipes* elsewhere. There is one recursive step for each integer for negative values, so large negative values are not recommended. Result is 1/0 for x=0 or negative integers. Gamma function is positive everywhere except (–1,0), (–3,–2), (–5,–4), ….

```
int gcomdiv(int m,int n);
```
     This always returns a positive number. If either input is 0, then 1 is returned.

```
float reflect(float x,float lo,float hi);
```
     Reflects x off of lo and hi as many times as needed until x is within [lo,hi]. Returns the reflected value of *x*. This uses recursion which is slow, so the routine could be sped up some. It is especially slow if x starts far below lo or far above hi. The function will hang if lo is not less than hi.

```
float inversefn(float (*fn)(float),float y,float x1,float x2,int n);
```
    inversefn returns the x value at which fn(x)=y, where x is between x1 and x2, which bracket the inverse value. It uses a bisection method with n steps and an algorithm similar to one described in *Numerical Recipes*. Returned value is on (x1,x2), which may be in either order, and has a maximum error of $2^{-(n+1)}$(x2–x1). If there are multiple solutions, the one that is found first will be returned. If there are no solutions, the returned value is nearly equal to the endpoint that is closer to the solution (with the same error as above).

```
double quadpts2paramsD(double *x,double *y,double *abc);
```
    For the quadratic $y=ax^2+bx+c$, this solves for the a, b, and c parameters from three (x,y) pairs. x is three known x values and y are the respective known y values. The result is returned in the three element vector abc, with a first, then b, and then c. The function uses the matrix $\mathbf{M}$ where $M_{ij} = x_i^{(2-j)}$, along with its determinant. The determinant is returned; a return value equal to 0 means that the parameters cannot be determined and abc was not calculated, and very small return values (positive or negative) indicate that there may be large numerical errors in abc. A zero determinant occurs if, and I think only if, two or more x values are identical.

```
double fouriersumD(double *a,double *b,int n,double l,double x);
```
    This calculates the sum of a Fourier series for a function which is periodic on $2l$ ($-l$ to l, or 0 to $2l$, or any other interval). The sum is defined by:

$$y = \frac{a_0}{2} + \sum_{j=1}^{n-1}\left[a_j \cos\left(\frac{j\pi x}{l}\right) + b_j \sin\left(\frac{j\pi x}{l}\right)\right]$$

    The parameters in the equation are exactly as they are in the function call. Note that b[0] is completely ignored and that the sum goes to $n-1$ rather than to n, where the latter is the standard math convention.

```
void SetHillParam(float *hp,float a,float e,float n);
```
    Sets vector of Hill function parameters called hp to a, e, and n. No math or checking is done, just assignment of a, e, and n to the 3-element hp vector, which needs to be pre-allocated. For additional Hill function details, see my MSI notes.

```
float HillFn(float *hp,float x);
```
    Hill function. hp is 3-element vector of Hill function parameters with values A, E, and n, respectively, and x is the variable. The returned value follows the Hill equation:

$$y = A\frac{x^n}{E^n + x^n}$$

```
void HillFnCompose(float *hp1,float *hp2,float *hp12);
```

Hill functions H_1 and H_2 are defined by parameters hp1 and hp2. Their composition is $f(x) = H_2(H_1(x))$, which is not a Hill function but is similar to one. The Hill function that shares the same maximum amplitude, x-value at half maximum, and log-slope at half-maximum as $f(x)$ has parameters hp12, which are calculated and returned by this function.

void HillFnComposeNF1(float *hp1,float *hp2,float *hpf1,float *hpf12);
Composition of Hill functions hp1 and hp2, including negative feedback from 2 to 1, to yield a "close" Hill function with parameters hpf12. The reaction mechanism is that the activated product of reaction 2 is a reactant for the inactivation of reaction 1. Either, both, or neither hpf1 and hpf2 are returned; for one to be returned send in a non-NULL address. It is assumed, but not checked, that both input n values (hp1[2] and hp2[2]) are equal to 1. This function could be generalized to other input n values, but that hasn't been done yet.